Neurocomputing 177 (2016) 136–146

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Orthogonal component analysis: A fast dimensionality reduction algorithm

Tao Zhu [a], Ye Xu [b,1], Furao Shen [a,*], Jinxi Zhao [a]

[a] National Key Laboratory for Novel Software Technology, Department of Computer Science and Technology, Nanjing University, Nanjing, PR China
[b] 6211 Sudikoff Lab, Dartmouth College, Hanover, NH 03755, USA

ARTICLE INFO

Article history:
Received 16 June 2015
Received in revised form
13 September 2015
Accepted 3 November 2015
Communicated by Feiping Nie
Available online 22 November 2015

Keywords:
Dimensionality reduction
Orthogonal component
Intrinsic dimension estimation
High-speed component learning

ABSTRACT

Most existing dimensionality reduction algorithms have two disadvantages: their computational cost is high and they cannot estimate the intrinsic dimension of the original dataset by themselves. To deal with these problems, in this paper we propose a fast linear dimensionality reduction method named Orthogonal Component Analysis (OCA). While avoiding solving eigenproblem and matrix inverse problem, OCA successfully achieves high-speed orthogonal component extraction. By proposing an adaptive threshold scheme, OCA is able to estimate the dimension of the feature space automatically. Meanwhile, the algorithm is guaranteed to be numerical stable. In the experiments, OCA is compared with several typical dimensionality reduction algorithms. The experimental results demonstrate that as a universal algorithm, OCA is efficient and effective.

1. Introduction

The "curse of dimensionality" is a serious problem for the machine learning, data mining and pattern recognition tasks that involve high-dimensional data [1]. Dimensionality reduction (DR) is an important strategy for addressing this problem. By reducing the dimensionality, we obtain a reduced representation of input data that is much smaller in volume. Meanwhile we can obtain the same (or almost the same) analytical result. Therefore, such techniques have been widely applied in image recognition [2,3], text analysis [4,5], image retrieval [6,7], hand motion classification [8], and gene selection [9].

Dimensionality reduction algorithms adopt mathematical transformation to convert the original high-dimensional data space into lower-dimensional feature space. Generally speaking, they can be categorized into two classes: linear DR algorithms and nonlinear DR algorithms.

Linear algorithms convert original high-dimensional data to low-dimensional subspace by linear transformation. Suppose $\boldsymbol{x} \in \mathbb{R}^{D\times 1}$ and $\boldsymbol{y} \in \mathbb{R}^{d\times 1}$ are the original high-dimensional data and the obtained low-dimensional representation in feature subspace respectively, the problem of linear DR can be transformed to calculating $\boldsymbol{B} = [\boldsymbol{b_1}, \boldsymbol{b_2}, \ldots, \boldsymbol{b_d}] \in \mathbb{R}^{D\times d}$ whose columns are the desired components or basis. There are a great number of typical linear DR algorithms, such as principal component analysis (PCA) [10], linear discriminant analysis (LDA) [11], independent component analysis (ICA) [12], random projection [13], and locality preserving projection (LPP) [14]. Among the above-mentioned algorithms, PCA and random projection generate orthogonal basis. As orthogonality is usually desired for basis of a subspace, many algorithms are improved by introducing orthogonality constraints, for example, orthogonal LDA (OLDA) [15], orthogonal locality preserving projection (OLPP) [16] and orthogonal projective nonnegative matrix factorization (OPNMF) [17].

Compared with linear DR algorithms, recently nonlinear algorithms have attracted great attention and many literatures demonstrate that nonlinear methods are better in capturing complex relationships of the data [18,19]. For nonlinear DR, manifold learning has been the main focus and most popular nonlinear algorithms include multidimensional scaling (MDS) [20], ISOMAP [21], self-organizing map (SOM) [22], locally linear embedding (LLE) [23], Laplacian eigenmap (LE) [24], local tangent space alignment (LTSA) [25], etc. However, manifold learning methods are not always preferable for data representation [26]. To determine the adjacency relation, manifold learning methods usually need extra prior knowledge (such as class labels and pairwise constraints) [27]. Moreover, on arbitrary or noisy data, the performance of manifold learning is problematic [28]. On the other hand, by employing kernel technology, many existing linear

* Corresponding author.
E-mail addresses: tao144@gmail.com (T. Zhu), ye@cs.dartmouth.edu (Y. Xu), frshen@nju.edu.cn (F. Shen), jxzhao@nju.edu.cn (J. Zhao).
[1] This work was done when the author was in Nanjing University.

http://dx.doi.org/10.1016/j.neucom.2015.11.012
0925-2312/

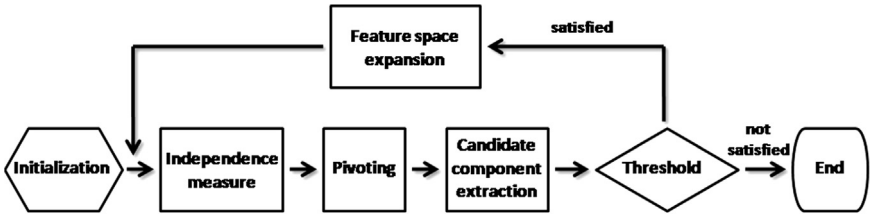

Fig. 1. The flowchart of OCA. In the beginning, the feature space is initialized as a zero-dimensional space. It expands during the learning process. According to the calculated independence degree, each time OCA extracts a new candidate component from the selected data vector. With the help of the threshold, OCA only accepts candidate component when it is necessary. Once the threshold cannot be satisfied, the algorithm stops and the low-dimensional feature space is determined.

algorithms are able to extend themselves to nonlinearity by projecting the data into a higher-dimensional feature space temporarily. KPCA [29] is the most famous and widely used algorithm among them.

Unfortunately, classical DR algorithms have several disadvantages: most of them involve eigenproblem or matrix inverse problem. Solving these problems is time-consuming. What's worse, it is likely for those algorithms to meet ill-conditioned problem in practical applications. Furthermore, although all these algorithms assume that the data of interest lie on or near certain low-dimensional linear subspace or non-linear manifold in the original high-dimensional space, few of them are able to estimate the dimension of the subspace or manifold by themselves. Usually users have to predetermine the target dimension empirically. However, an improperly determined target dimension may cause serious problems.

Obviously, how to overcome the deficiencies of the existing algorithms is a long-standing problem worth studying. In this paper we focus on linear dimensionality reduction. Linear dimensionality reduction can be viewed as a special case of common dimensionality reduction, they are fundamental and many basic ideas of nonlinear algorithms are derived from them.

In the past two decades, many new linear dimensionality reduction algorithms have been proposed [30–32]. Specially, some algorithms are designed to improve the speed of classical algorithms [33–35]; some algorithms are able to estimate the target dimension d automatically [36,37]. However, to the best of our knowledge, few of the existing algorithms are able to achieve high-speed component learning while automatically determining the target dimension with little prior knowledge and few additional calculations.

In this paper, we propose a fast dimensionality reduction algorithm named Orthogonal Component Analysis (OCA). Many researches have indicated that compared with non-orthogonal components, orthogonal components are more desired in dimensionality reduction [38,39,40]. Therefore, as a fast algorithm, OCA employs Gram–Schmidt (GS) algorithm [41] to calculate the orthogonal candidate components. Obviously, GS has the excellent property of low computational complexity. However, classical GS has a serious drawback: it is numerically unstable. Fortunately, researchers have paid great attention to solve this problem and fruitful achievements have been made [42]. Based on the previous studies, in OCA, data selection is processed to make sure that the vectors to be orthogonalized are fairly independent. Then, we propose an adaptive threshold scheme that is the key contribution of the OCA algorithm. With the help of the threshold scheme, OCA only accepts the ideal candidate components, and thus it is able to automatically determine the proper number of components. Although the threshold scheme is simple, it needs no prior knowledge and little extra computational cost. More importantly, it makes OCA an efficient and effective algorithm: the dimension of feature space is automatically determined; the low-dimensional representation of original data is quality-assured; the numerical stability of OCA is ensured while its computational complexity is low.

We organize the rest of this paper as follows. The proposed OCA algorithm is introduced in detail in Section 2 and then some theoretical discussions about the algorithm are given in Section 3. In Section 4, experiments are taken and results are reported. Finally in Section 5, the paper is concluded.

2. Orthogonal component analysis (OCA)

As mentioned in Section 1, the targets of the proposed OCA are

- Learning orthogonal components for original data with low computational cost and high stability;
- automatically determining the number of components.

We give the flowchart of the OCA algorithm in Fig. 1. Assume $\{\boldsymbol{x}_i\}_{i=1}^{N}$ are the N input data from which we are going to extract orthogonal components. OCA initializes feature space basis $\mathcal{B}=\varnothing$, and the number of learned components $k=0$.

In each iteration of the loop, the proposed OCA always tries to extract a new component from the most ideal pattern by data selection.

Firstly, OCA measures the degree of independence between the data vectors and the feature space S.

Based on the independence degree that is calculated in every iteration, OCA selects the data that has the strongest independence with the feature space as the pivot element. Then OCA calculates a candidate component from it.

An adaptive threshold policy is employed to automatically decide whether to accept the candidate component. If it is accepted, feature space S expands and the learning process continues. If the threshold condition is not satisfied, we will stop the whole algorithm and obtain the final feature space.

The pivoting procedure allows the OCA algorithm to process successfully and possibly to reduce round-off error. The threshold policy helps OCA to avoid ceaselessly learning N candidate components from the N input data. It not only reduces the computational load, but also ensures the numerical orthogonality between the learned components.

There are a great number of researches about the selection technologies in DR [43–46]. For OCA, it selects sample data that are fairly independent, and as a forward selection method, OCA enjoys the advantage of low computational complexity. Note that the concept of GS with pivoting has been employed by many algorithms for selection purposes [47–49], they do not extract orthogonal components. Especially, in [49], the authors theoretically proved that under certain assumptions, this family of algorithms is robust under any small perturbations of the input data. However, these algorithms are not able to automatically estimate the number of extracted vectors. With the help of the proposed adaptive threshold policy, OCA not only ensures the numerical

orthogonality among the extracted components but also determines the target dimension automatically.

In the rest of the section, we represent the proposed OCA algorithm in detail.

2.1. Independence measure

The feature space S is initialized as a zero-dimensional space. By adding newly learned components, feature space S expands. Once all the data are dependent with S, OCA stops learning and S achieves a steady state. Therefore, the intrinsic dimension of the original dataset is estimated.

During the component extracting process, the blind pursuit of finding out all the orthogonal components of the dataset without component selecting will not only increase the computational burden but also poison the data analysis process.

Take the famous $n \times n$ Hilbert Matrix $\boldsymbol{H}(n)$ [50] for example, its entry in the ith row and jth column is $\boldsymbol{H}_{ij} = 1/(i+j-1)$. Thereby, $\boldsymbol{H}(n)$ must be a nonsingular matrix whose rank is n. It means that, theoretically speaking, the dimension of the space that is spanned by $\boldsymbol{H}(n)$'s column vectors should be n. However, in a case study, we validate that there are only about 18 linear independent basis vectors in the space spanned by $\boldsymbol{H}(100)$'s 100 column vectors in Matlab. If we set up the dimension of the space as 100, due to the round-off error, many learned basis vectors of this space are numerically nonorthogonal and this too large predetermined dimension increases the computational load of the algorithm. The main reason for this may be that the variations of the dataset in some directions are so little that they can be omitted. To discard reluctant information and build robust learning model, we have to measure the numerical independence between the input data and the learned components.

Suppose we have learned k orthonormal vectors $\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_k$ that span space S. For each pattern $\boldsymbol{x}_j$, linear dependence theorem [51] indicates that we can use the projection distance $\boldsymbol{x}_j$ onto space S to measure the numerical independence between them:

$$\| span\{\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_k\} - \boldsymbol{x}_j \|_2 \tag{1}$$

or equivalently

$$\min_{\boldsymbol{\alpha}} \| \boldsymbol{B}_k \boldsymbol{\alpha} - \boldsymbol{x}_j \|_2 \tag{2}$$

where $\boldsymbol{\alpha}$ is the coordinate vector, $\boldsymbol{B}_k = [\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_k] \in \mathbb{R}^{D \times k}$ is the basis matrix.

Obviously, when $\boldsymbol{\alpha} = \boldsymbol{B}_k^\top \boldsymbol{x}_j$, the solution of (2) is obtained. Therefore, the independence degree between data vector $\boldsymbol{x}_j$ and the feature space S can be measured by

$$\left\| \boldsymbol{x}_j - \sum_{i=1}^{k} \boldsymbol{b}_i \boldsymbol{b}_i^\top \boldsymbol{x}_j \right\|_2 \tag{3}$$

We define

$$\boldsymbol{x}_j^{(k)} = \boldsymbol{x}_j - \sum_{i=1}^{k} \boldsymbol{b}_i \boldsymbol{b}_i^\top \boldsymbol{x}_j \tag{4}$$

Therefore, as shown in Fig. 2, if the value of $\left\| \boldsymbol{x}_j^{(k)} \right\|_2$ is small, $\boldsymbol{x}_j$ is strongly dependent on the learned k components. It is unsuitable for the algorithm to accept the $(k+1)$th component $\boldsymbol{b}_{k+1}$ extracted from it.

2.2. Pivoting

Pivoting is a powerful technique in matrix computation that can decrease the round-off error and improve the stability of algorithm [52]. The pivot element is an element that is selected firstly by an algorithm to do certain calculations. Finding this element is called pivoting. It has been used in Gaussian elimination process, Householder reflection, Cholesky factorization, GS algorithm and so on. Especially, pivoting can significantly improve numerical stability for GS algorithm.

In the $(k+1)$th iteration, OCA selects the data vector $\boldsymbol{x}_{j_{k+1}^*}$ that has the strongest independence with the current feature space S. As discussed in Section 2.1, we obtain it by:

$$j_{k+1}^* = \arg\max_{1 \le j \le N} \left\| \boldsymbol{x}_j^{(k)} \right\|_2 \tag{5}$$

Then, $\boldsymbol{x}_{j_{k+1}^*}^{(k)}$ is the pivot element selected by OCA.

We define

$$r_{k+1,k+1} = \left\| \boldsymbol{x}_{j_{k+1}^*}^{(k)} \right\|_2 \tag{6}$$

If $r_{k+1,k+1} = 0$, it means the whole dataset $\{\boldsymbol{x}_i\}_{i=1}^N$ can be perfectly represented by the learned k components and there is no need to extract any new component. Otherwise, OCA calculates the $(k+1)$th candidate component:

$$\boldsymbol{b}_{k+1} = \frac{\boldsymbol{x}_{j_{k+1}^*}^{(k)}}{r_{k+1,k+1}} \tag{7}$$

Therefore, theoretically speaking, the candidate component $\boldsymbol{b}_{k+1}$ and the learned k components $\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_k$ are mutually orthogonal. Unfortunately, in computer, the calculated approximation of a number and its exact mathematical value are different. The round-off may accumulate through a sequence of calculations. Sometimes significant error has accumulated and it may completely destroy the mutually orthogonality between these components. Though to some extent, pivoting can prevent the problem. A threshold is still needed to rule out unsuitable $\boldsymbol{b}_{k+1}$ that is calculated after pivoting. Only if $r_{k+1,k+1} \ge T$, $\boldsymbol{b}_{k+1}$ will be accepted and added into basis $\mathcal{B}$; otherwise, OCA discards $\boldsymbol{b}_{k+1}$ and stops learning new components.

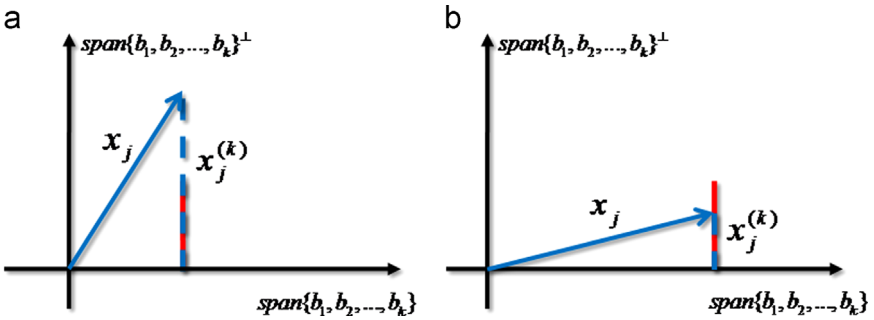

Fig. 2. The vector $\boldsymbol{x}_j$ is projected onto two orthogonal space: $S = span\{\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_k\}$ and $span\{\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_k\}^\perp$. The independence degree between $\boldsymbol{x}_j$ and S can be measured by $\| \boldsymbol{x}_j^{(k)} \|_2$. (a) When $\| \boldsymbol{x}_j^{(k)} \|_2$ is larger than the threshold (red line), OCA may learn a new component from $\boldsymbol{x}_j$; (b) otherwise, OCA will not learn component from it. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

Theoretically speaking, if the extracted $\{\boldsymbol{b}_i\}_{i=1}^{k}$ are mutually orthogonal, we have $\left\|\boldsymbol{x}_j^{(k)}\right\|_2^2 \le \left\|\boldsymbol{x}_j^{(k-1)}\right\|_2^2$. Or more precisely,

$$\left\|\boldsymbol{x}_j^{(k)}\right\|_2^2 = \left\|\boldsymbol{x}_j^{(k-1)}\right\|_2^2 - \left(\boldsymbol{b}_k^{\top}\boldsymbol{x}_j\right)^2 = \left\|\boldsymbol{x}_j^{(k-1)}\right\|_2^2 - \left(\boldsymbol{b}_k^{\top}\boldsymbol{x}_j^{(k-1)}\right)^2 \tag{8}$$

2.3. Threshold policy

OCA estimates the intrinsic dimension of the original dataset by learning the necessary components. Once newly extracted component has been accepted, feature space S expands. When there is no data that are highly independent with S, we believe that S can represent the whole dataset well and the current number of components is the estimated intrinsic dimension of the original dataset. Therefore, the obtained feature space is completely data-dependent.

Instead of employing an absolute threshold that has to be determined empirically, we propose an adaptive threshold policy that can adjust according to the original dataset and the current $\mathcal{B}$. Intuitively, if the threshold can automatically adjust itself according to the current condition, dimension estimation will be more flexible and the influence of improper initial threshold setting will be reduced.

The adjustment of feature subspace S is influenced by two aspects. On one hand, we hope S to contain as much information as possible, thus S wants to expand. On the other hand, as a dimensionality reduction task, it is natural that the compression ratio (the number of extracted components divided by the dimension of input data) is the smaller the better. Therefore, when feature subspace S is small, it tends to learn more information from input data; when S is large, preventing its blind expansion is more important than learning new knowledge. Expansion or maintenance, OCA should automatically take balance between them by the threshold T. Once the balance is achieved, the target dimension is determined. As a result of that, if the compression ratio (k/D) is small currently, it is imperative to accept potential basis vectors and vice versa.

As described above, the proposed threshold T should satisfy the following two constraints: (i) The strong independence between components should be ensured. (ii) The difficulty of accepting components increases with the expansion of the feature space S.

To satisfy these two constraints, we propose an adaptive threshold policy as: if and only if

$$r_{k+1,k+1} \ge f\left(\frac{k}{D}\right) C \tag{9}$$

we believe that extracting $\boldsymbol{b}_{k+1}$ is necessary and accept it as a new component. Otherwise, $\boldsymbol{b}_{k+1}$ is discarded and $S = span\{\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_k\}$ is the final feature space. Meanwhile, $d=k$ is the estimated intrinsic dimension of the original dataset.

$f(\omega)$ is the threshold function. For convenience, we impose a constraint on $f(\omega)$: it is a strictly increasing function and $0 \le f(\omega) \le 1$ when $0 \le \omega \le 1$. C is a parameter automatically learned from the original dataset and it does not change in the learning process. We hope it satisfies $C \ge r_{k+1,k+1}$.

Based on the threshold policy, OCA can select the training data that has little dependence on learned components and extract data-dependent numerically orthogonal component from it rapidly. In this way, the accumulation of round-off error in these calculations can be greatly avoided. Therefore, if components $\boldsymbol{b}_{i_1}$ and $\boldsymbol{b}_{i_2}$ are obtained in different iterations of the OCA algorithm, they are strictly orthogonal to each other.

Before giving the setting of C, we represent the following conclusion:

Theorem 1. *Suppose OCA finally learns d mutually orthogonal components from dataset $\mathcal{X} = \{\boldsymbol{x}_j\}_{j=1}^{N}$. The values of $\{r_{i,i}\}_{i=1}^{d+1}$ achieved in the iterations of OCA are monotonically non-increasing, i.e. if $1 \le k_1 \le k_2 \le d+1$ then $r_{k_1,k_1} \ge r_{k_2,k_2}$.*

Proof. According to (8), we have

$$r_{k_2,k_2} = \|\boldsymbol{x}_{j_{k_2}^*}^{(k_2-1)}\|_2 \le \|\boldsymbol{x}_{j_{k_2}^*}^{(k_1-1)}\|_2 \le \|\boldsymbol{x}_{j_{k_1}^*}^{(k_1-1)}\|_2 = r_{k_1,k_1} \tag{10}$$

□

We know that $r_{k+1,k+1} \le r_{1,1}$. $r_{1,1}$ is the length of the longest data vector in the original dataset. Therefore, we set $C = r_{1,1}$ and rewrite (9) as

$$\frac{r_{k+1,k+1}}{r_{1,1}} \ge f\left(\frac{k}{D}\right) \tag{11}$$

Given a certain dataset, during the learning process of OCA, as k increases, the value of $f(k/D)$ is strictly increasing and $r_{k+1,k+1}/r_{1,1}$ is monotonically non-increasing. Once (11) cannot be met, there will never be a candidate component that can satisfy it any more. As a result of that, OCA can stop learning at that time.

According to the above description, the proposed threshold policy guarantees that the obtained subspace will not expand ceaselessly and "Cauchy criterion for convergence" is satisfied. Suppose $\{a_1, a_2, \ldots, a_D, a_{D+1}, \ldots\}$ is a sequence generated as follows: if $r_{i,i} \ge f((i-1)/D) r_{1,1}$ (it implies $i \le D$), then $a_i = 1$; otherwise, $a_i = 0$. Therefore, there is a number d, for all $m > d$, $a_m = 0$. We let $s_j = \sum_{i=1}^{j} a_i$ and s_j equals to the dimension of the subspace obtain by OCA at time j. Obviously, $|s_{m+p} - s_m| = |a_{m+1} + a_{m+2} + \cdots + a_{m+p}| = 0$ holds for all $m \ge d$ and $p \ge 1$.

There are a great deal of research work about intrinsic dimension estimation [53]. According to Fukunaga's definition [54], a dataset $\mathcal{X} \subseteq \Omega^D$ is said to have intrinsic dimension equal to d_0 if its elements lie entirely within a d_0-dimensional subspace (where $d_0 < D$). With the help of the proposed threshold policy, OCA guarantees that for each original data vector, the distance from it to the obtained d-dimensional subspace is bounded. Therefore, we deem that all the original data distribute on or near the obtained d-dimensional subspace and d is the estimated intrinsic dimension of dataset $\mathcal{X}$.

In summary, OCA employs automatic threshold to enhance the stability of the algorithm and estimate the dimension of feature subspace. Though the employed strategy is very simple and we cannot promise that the target dimension d estimated by OCA is optimal. This simple strategy has the advantages of high-efficiency and self-adaption. d is obtained automatically and data-dependently. Moreover, no additional prior knowledge is needed and the computational cost is very low. Based on the threshold policy, OCA can select those training data that have little dependence on learned components and extract data-dependent orthogonal components from them rapidly.

2.4. Summary of OCA

Given the input data $\{\boldsymbol{x}_j\}_{j=1}^{N}$, OCA continuously extracts d orthogonal components $\{\boldsymbol{b}_i\}_{i=1}^{d}$. Here, d is automatically determined by OCA with no prior knowledge. Furthermore, OCA outputs original data's low-dimensional representations $\{\boldsymbol{y}_j\}_{j=1}^{N}$.

In the beginning, basis $\mathcal{B}$ of the feature space S is initialized as an empty set. Accordingly, the dimensionality of S is $k=0$. For each $\boldsymbol{x}_j$, $\|\boldsymbol{x}_j^{(0)}\|_2^2 = \|\boldsymbol{x}_j\|_2^2$ is calculated. Then, OCA starts to learn new components in each iteration and expand feature space S.

Suppose in the $(k+1)$ iteration, the j_{k+1}^{*th} data of the original dataset is chosen as the $(k+1)$th pivot element. (Usually, pivoting may be followed by an interchange of elements to bring the pivot element to a fixed position. However, the vector elements

movement would cost too much time. Instead, for convenience, we just keep track of the N-element permutations.) We calculate the new candidate component $\boldsymbol{b}_{k+1}$ from it.

If the threshold is satisfied, we accept $\boldsymbol{b}_{k+1}$ and add it to basis $\mathcal{B}$. Then, for each original data vector $\boldsymbol{x}_j$, we calculate $y_{j,k+1}=\boldsymbol{b}_{k+1}^{\top}\boldsymbol{x}$ that is the $(k+1)$th entry of $\boldsymbol{x}_j$'s low-dimensional representation. Note that if $\boldsymbol{x}_j$ has been selected as the pivot element in the previous iterations, then $\boldsymbol{b}_{k+1}^{\top}\boldsymbol{x}_j=0$. Base on this conclusion, the calculations can be simplified a little.

After that, we update $k=k+1$. By (8), $\left\|\boldsymbol{x}_j^{(k)}\right\|_2^2$ is obtained and we continue to extract the next component in a new iteration.

Once $r_{k+1,k+1}/r_{1,1}<f(k/D)$, we believe the current feature space S can represent the original dataset well and there is no need to learn new components. Therefore, the candidate component $\boldsymbol{b}_{k+1}$ is discarded and the learning process stops. Then, we obtain the final target dimensionality $d=k$. OCA outputs the final results: basis $\mathcal{B}$, target dimensionality d and $\{\boldsymbol{y}_j\}_{j=1}^{N}$ ($\boldsymbol{y}_j=(y_{j,1}, y_{j,2},\ldots,y_{j,d})^{\top}$).

According to the description above, we give the detailed OCA in Algorithm 1. The whole algorithm generates numerically orthogonal basis vectors and determines proper dimension for the feature subspace. Meanwhile, the low-dimensional representation of the original data is obtained. In addition, OCA applies data selection scheme and obviates the solution of eigenproblem and matrix inverse problem. Therefore, it is an efficient algorithm.

Algorithm 1. Orthogonal component analysis algorithm.

Input: Dataset $\{\boldsymbol{x}_j\}_{j=1}^{N}(\forall 1\le j\le N, \boldsymbol{x}_j\in\mathbb{R}^{D\times 1})$.

1: Define $\forall 1\le j\le N$, $\boldsymbol{x}_j^{(0)}=\boldsymbol{x}_j$.
2: Initialize the basis $\mathcal{B}=\varnothing$ and the dimension $k=0$.
3: Fine the $(k+1)$th pivot element $\boldsymbol{x}_{j_{k+1}^*}$ by (5).
4: Compute $r_{k+1,k+1}$ and $\boldsymbol{b}_{k+1}$ by (6) and (7) respectively.
5: **if** $\frac{r_{k+1,k+1}}{r_{1,1}}\ge f\left(\frac{k}{D}\right)$ **then**
6: Accept $\boldsymbol{b}_{k+1}$ as a basis vector and add it into $\mathcal{B}$.
7: Set $y_{j,k+1}=\boldsymbol{b}_{k+1}^{\top}\boldsymbol{x}_j^{(k)}$, for $j=1:N$.
8: Update $\boldsymbol{x}_j^{(k+1)}=\boldsymbol{x}_j^{(k)}-y_{j,k+1}\boldsymbol{b}_{k+1}$, for $j=1:N$.
9: Update dimension $k\leftarrow k+1$.
10: Goto step 3 to continue the learning process.
11: **else**
12: Set $d=k$.
13: Stop the algorithm.
14: **end if**

Output $\mathcal{B}$, d and $\{\boldsymbol{y}_j\}_{j=1}^{N}$.

3. Analysis

Firstly, there are some obvious conclusions: with the help of pivoting and the adaptive threshold scheme, if there are data vectors that are strongly independent to the learned basis, they will not be neglected. This ensures that the number of components finally learned by OCA will never be improperly small.

Secondly, as k (the number of learned components) increases, for every data vector in $\mathcal{X}$, the dependence between it and the feature space becomes stronger and stronger. Meanwhile the threshold is stricter and stricter. Therefore, it is certain that as the feature space expands, it is more and more difficult for OCA to accept new components. Theoretically, OCA will never obtain more than D components.

Moreover, to theoretically demonstrate that the proposed OCA algorithm is efficient and effective, we analyze it in three aspects: (i) reconstruction cost; (ii) time complexity and (iii) stability.

3.1. Reconstruction cost

Assume a map $f:\boldsymbol{x}\in\mathbb{R}^D\rightarrow\boldsymbol{y}\in\mathbb{R}^d, d\le D$, an important criterion for the performance of this map is whether the original high-dimensional $\boldsymbol{x}$ can be well approximated by $\tilde{\boldsymbol{x}}$ that is reconstructed by the projected low-dimensional $\boldsymbol{y}$.

Most existing algorithms cannot determine the target dimension d by themselves. If user sets the target dimension improperly small, these methods will suffer from the problem of poor approximation. Moreover, there are few dimensionality reduction algorithms that can give a definite upper bound of approximation error for each data $\boldsymbol{x}$.

Comparing with these methods, a key factor establishes the advantage of OCA: the adaptive threshold scheme. It not only automatically determines the target dimension k, but also bounds the approximation error for each $\boldsymbol{x}$.

For OCA, we can obtain the following conclusion:

Theorem 2. *For each data $\boldsymbol{x}$ processed by OCA, there is a determined upper bound for its approximation error:*

$$\|\boldsymbol{x}-\tilde{\boldsymbol{x}}\|_2<r_{1,1}f\left(\frac{d}{D}\right) \tag{12}$$

Proof. d is the final target dimension. The low-dimensional representation of $\boldsymbol{x}$ is wrote as a d-dimensional vector: $\boldsymbol{y}=(y_1,y_2,\ldots,y_d)^{\top}$. We have $\|\boldsymbol{x}-\tilde{\boldsymbol{x}}\|_2=\|\boldsymbol{x}-\sum_{i=1}^{d}y_i\boldsymbol{b}_i\|_2=\|\boldsymbol{x}^{(d)}\|_2$. According to the threshold policy of OCA, if $\|\boldsymbol{x}^{(d)}\|_2\ge r_{1,1}f(d/D)$, new basis will be extracted and the feature subspace expands. The dimension of $\boldsymbol{y}$ should be more than d, a contradiction.□

As mentioned above, it is obvious that when d orthogonal components have been obtained, there are at least d data vectors in the original dataset can be perfectly reconstructed by these learned components.

Therefore, given a dataset $\mathcal{X}$, d orthogonal components are obtained by OCA. As described above, the lager the value of d is, the more the data in $\mathcal{X}$ can be perfectly approximated. The smaller the value of d is, the lower the upper bound of the approximation error for each data in $\mathcal{X}$ is restricted. This conclusion does not mean OCA will never output a non-ideal result whatever the dataset is, but it theoretically demonstrates the effectiveness of this algorithm to some extent.

3.2. Time complexity

The time complexity of OCA is determined by three parts: the pivoting procedure, the computation of candidate component, and the updating operation. Define N is the size of training set, D is the original dimension of input vector, and d is the number of learned components in feature space. In each iteration, the time complexity of finding the pivot element is $O(ND)$; for the computation of $\boldsymbol{b}_{k+1}$, the time complexity is $O(D)$; the updating procedure accounts for complexity of $O(ND)$. There are $d+1$ iterations in OCA. Thus time complexity of the proposed OCA is $O(NDd)$. To compare with OCA, we analyze the time complexity of traditional PCA,

Table 1
Time complexity of OCA, PCA, BPCA, CCIPCA, and FastICA.

Algorithm	OCA	PCA	BPCA	CCIPCA	FastICA
Time complexity	$O(N\,Dd)$	$O(D^3+ND^2+D^2d)$	$O(ND^3d)$	$O(N\,Dd)$	$O(hN\,Dd)$

BPCA [36], CCIPCA [33] (an online version of PCA) and FastICA [35] (a fast version of ICA) and list them in Table 1. For FastICA, h is the number of iterations.

OCA applies an iterative procedure, step by step, to learn the components. The calculations end in a finite number of steps. Unlike many other DR algorithms, OCA avoids solving eigenproblem and matrix inversion problem. Usually eigenproblem has to be solved by numerical computation that cannot be expected to terminate in a finite number of steps. Therefore, it is time consuming to solve eigenproblem. Moreover, sometimes due to the ill-conditioned matrix, the obtained solutions are not reliable, especially for the algorithms that aim to obtain the minimum eigenvalue solutions. For those methods involve matrix inversion, once the matrix is singular, they will fail to work. It also suffers from ill-conditioned problem. For FastICA, the number of iterations h is also almost impossible to predict. Sometimes a great number of iterations have to be executed before these algorithms can achieve a convergence.

According to Table 1, we obtain the conclusion that the time complexity of OCA is far smaller than that of PCA, BPCA, and FastICA. Table 1 also indicates that OCA runs as fast as online algorithm such as CCIPCA.

3.3. Stability of OCA

Without pivoting and threshold, the orthonormal component calculation process in Algorithm 1 can be seen as modified Gram–Schmidt (MGS). MGS yields a better computational procedure than Classical Gram–Schmidt (CGS), while its computational cost is very low (for example, it is about twice as efficient as Householder orthogonalization) [41]. However, [55] indicates that the $\boldsymbol{B}=[\boldsymbol{b_1}, \boldsymbol{b_2}, \ldots, \boldsymbol{b_d}]$ computed from data $\boldsymbol{A}=[\boldsymbol{x}_{j_1^*}, \boldsymbol{x}_{j_2^*}, \cdots, \boldsymbol{x}_{j_d^*}]$ by MGS satisfies

$$\boldsymbol{B}^\top \boldsymbol{B} = \boldsymbol{I} + \boldsymbol{E}\|\boldsymbol{E}\|_2 \approx \boldsymbol{u}\kappa_2(\boldsymbol{A}) \tag{13}$$

$\boldsymbol{I}$ is the unit matrix, $\boldsymbol{u}$ is the unit round-off, and $\kappa_2(\boldsymbol{A})$ is the 2-norm condition of matrix $\boldsymbol{A}$.

Obviously, for MGS and OCA, the smaller the $\|\boldsymbol{E}\|_2$ is, the better the accuracy is. To achieve small $\|\boldsymbol{E}\|_2$, the selected data $\boldsymbol{x}_{j_1^*}^*, \boldsymbol{x}_{j_2^*}^*, \ldots, \boldsymbol{x}_{j_d^*}^*$ should be fairly independent. That is precisely what can be ensured by OCA through employing the pivoting operation and an adaptive threshold scheme.

When $(d \geq 2)$, according to (13), OCA tries to make $\|\boldsymbol{E}\|_2$ as small as possible. To estimate the value of $\|\boldsymbol{E}\|_2$, we have to know the value of $\boldsymbol{A}$'s condition number $\kappa_2(\boldsymbol{A})$. However, $\kappa_2(\boldsymbol{A}) = \sigma_{max}(\boldsymbol{A})/\sigma_{min}(\boldsymbol{A})$ is difficult to calculate and it would be highly uneconomical to calculate it precisely in the fast component extraction process. Therefore, we find a substitute for it: we define $r_{1,1}/r_{d,d}$ as $\boldsymbol{A}$'s pseudo-condition number. Note that $\kappa_2(\boldsymbol{A})$ and $r_{1,1}/r_{d,d}$ have many similar properties.

Theorem 3. *In OCA, we have: (1) $\kappa_2(\boldsymbol{A})$ and $r_{1,1}/r_{d,d}$ have the same lower bound $\max\|\boldsymbol{x}_{j_k^*}\|_2/\min\|\boldsymbol{x}_{j_k^*}\|_2$; (2) As d increases, $\kappa_2(\boldsymbol{A})$ and $r_{1,1}/r_{d,d}$ are both monotonically nondecreasing; (3) If OCA obtains $\boldsymbol{A}' = [\boldsymbol{x}_1', \boldsymbol{x}_2', \ldots, \boldsymbol{x}_d']$ from another dataset, $\boldsymbol{x}_1', \boldsymbol{x}_2', \ldots, \boldsymbol{x}_d'$ are mutually orthogonal, and $r'_{1,1}/r'_{d,d}$ is $\boldsymbol{A}'$'s pseudo-condition number, then $r'_{1,1}/r'_{d,d} = \kappa_2(\boldsymbol{A}') = \|\boldsymbol{x}_1'\|_2/\|\boldsymbol{x}_d'\|_2$; (4) If the columns of $\boldsymbol{A}'$ also satisfy $\forall 1 \leq k \leq d$, $\|\boldsymbol{x}_k'\|_2 = \|\boldsymbol{x}_{j_k^*}\|_2$, then $\kappa_2(\boldsymbol{A}) \geq \kappa_2(\boldsymbol{A}')$ and $r_{1,1}/r_{d,d} \geq r'_{1,1}/r'_{d,d}$.*

Proof. Here, to save space, we only prove Property 1. Properties 2–4 can be easily proved.

According to the properties of eigenvectors,

$$\kappa_2(\boldsymbol{A}) = \frac{\sigma_{max}(\boldsymbol{A})}{\sigma_{min}(\boldsymbol{A})} = \frac{\sqrt{\lambda_{max}(\boldsymbol{A}^\top \boldsymbol{A})}}{\sqrt{\lambda_{min}(\boldsymbol{A}^\top \boldsymbol{A})}} = \frac{\sqrt{\max_{\boldsymbol{p}\in\mathbb{R}^d, \|\boldsymbol{p}\|=1}(\boldsymbol{p}^\top \boldsymbol{A}^\top \boldsymbol{A}\boldsymbol{p})}}{\sqrt{\max_{\boldsymbol{q}\in\mathbb{R}^d, \|\boldsymbol{q}\|=1}(\boldsymbol{q}^\top \boldsymbol{A}^\top \boldsymbol{A}\boldsymbol{q})}} \tag{14}$$

Obviously,

$$\max_{\boldsymbol{p}\in\mathbb{R}^d, \|\boldsymbol{p}\|=1} \boldsymbol{p}^\top \boldsymbol{A}^\top \boldsymbol{A}\boldsymbol{p} \geq \boldsymbol{e}_{max}^\top \boldsymbol{A}^\top \boldsymbol{A}\boldsymbol{e}_{max} = \max\|\boldsymbol{x}_{j_k^*}\|_2 \tag{15}$$

$$\min_{\boldsymbol{q}\in\mathbb{R}^d, \|\boldsymbol{q}\|=1} \boldsymbol{q}^\top \boldsymbol{A}^\top \boldsymbol{A}\boldsymbol{q} \leq \boldsymbol{e}_{min}^\top \boldsymbol{A}^\top \boldsymbol{A}\boldsymbol{e}_{min} = \min\|\boldsymbol{x}_{j_k^*}\|_2 \tag{16}$$

Therefore,

$$\kappa_2(\boldsymbol{A}) \geq \frac{\max\|\boldsymbol{x}_{j_k^*}\|_2}{\min\|\boldsymbol{x}_{j_k^*}\|_2} \tag{17}$$

On the other hand, according to the data selection scheme of OCA,

$$r_{1,1} = \max\|\boldsymbol{x}_{j_k^*}\|_2 = \|\boldsymbol{x}_{j_1^*}\|_2,\ r_{d,d} = \min\|\boldsymbol{x}_{j_k^*}^{k-1}\|_2 \leq \min\|\boldsymbol{x}_{j_k^*}\|_2 = \|\boldsymbol{x}_{j_d^*}\|_2 \tag{18}$$

Therefore,

$$\frac{r_{1,1}}{r_{d,d}} \geq \frac{\max\|\boldsymbol{x}_{j_k^*}\|_2}{\min\|\boldsymbol{x}_{j_k^*}\|_2} \tag{19}$$

□

For convenience, we use $r_{1,1}/r_{d,d}$ instead of $\kappa_2(\boldsymbol{A})$ in the estimation of $\|\boldsymbol{E}\|_2$. Thanks to the threshold function (11), we obtain:

$$\|\boldsymbol{E}\|_2 \approx \frac{r_{1,1}}{r_{d,d}}\boldsymbol{u} \leq \frac{1}{f\left(\frac{d-1}{D}\right)}\boldsymbol{u} \tag{20}$$

As threshold function $f(d/D)$ is a strictly increasing function, we can also have $1/f((d-1)/D)\boldsymbol{u} \leq 1/f(1/D)\boldsymbol{u}$. Moreover, in OCA, as d increases, $\|\boldsymbol{E}\|_2$ may increase. However, the upper bound of $\|\boldsymbol{E}\|_2$'s estimated value is strictly decreasing. In that way, OCA controls the influence of the round-off error and the ill-conditioned problem can be prevented.

Therefore, we can believe that the proposed OCA algorithm is numerical stable.

4. Experiment

To evaluate the performance of OCA, we conduct experiments on synthetic dataset and some widely used real-world datasets. All the experiments in this paper are run in MATLAB 7 on WinXP, CPU 2.4 GHz, memory 2 GB.

4.1. Experiments on synthetic data

4.1.1. Evaluation of orthogonality loss

To evaluate the orthogonality of components by OCA, we conduct the following experiment that is similar with the experiment in [56]: we generate 50×10 matrix $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^\top$. Therein, $\boldsymbol{U}$ and $\boldsymbol{V}$ are randomly generated 50×50 and 10×10 orthogonal matrixes respectively; $\boldsymbol{D} = diag(1, 10^{-1}, \ldots, 10^{-r+1}, \underbrace{1, \ldots, 1}_{10-r})$ whose size is 10×10. Therefore, $\boldsymbol{A}$'s condition number is $\kappa_2(\boldsymbol{A}) = 10^{r-1}$. Without the threshold, OCA can been seen as MGS with pivoting. In this section, we employ OCA without threshold to extract orthogonal basis from the columns of $\boldsymbol{A}$ and obtained $\boldsymbol{B_o}$. $\|\boldsymbol{I_{10}} - \boldsymbol{B_o}^\top \boldsymbol{B_o}\|_2$ is employed to represent $\boldsymbol{B_o}$'s loss of orthogonality. $\boldsymbol{I_{10}}$ is 10×10 identity matrix. Obviously, the smaller $\|\boldsymbol{I_{10}} - \boldsymbol{B_o}^\top \boldsymbol{B_o}\|_2$ is, the more orthogonal the produced $\boldsymbol{B_o}$ are. As a comparison, we also employ CGS to extract $\boldsymbol{B_c}$ and calculate $\|\boldsymbol{I_{10}} - \boldsymbol{B_c}^\top \boldsymbol{B_c}\|_2$. In Table 2, $\boldsymbol{B_c}$ and $\boldsymbol{B_o}$'s losses of orthogonality are compared when $\kappa_2(\boldsymbol{A})$ varies. The listed results are the average of 10 replicates.

We can find that as $\kappa_2(\boldsymbol{A})$ increases, for both CGS and OCA, their generated basis depart from orthogonality. However, OCA

Table 2
Comparison of orthogonality loss between the basis obtained by CGS and OCA.

$\kappa_2(\boldsymbol{A})$	$\|\boldsymbol{I_{10}}-\boldsymbol{B_c}^{\top}\boldsymbol{B_c}\|_2$	$\|\boldsymbol{I_{10}}-\boldsymbol{B_o}^{\top}\boldsymbol{B_o}\|_2$
1	6.7166×10^{-16}	5.7211×10^{-16}
10	3.0531×10^{-15}	2.7331×10^{-15}
100	4.2924×10^{-14}	2.2461×10^{-14}
10^3	3.9160×10^{-12}	1.7168×10^{-13}
10^4	6.8766×10^{-10}	1.3612×10^{-12}
10^5	1.8130×10^{-7}	1.3280×10^{-11}
10^6	3.3405×10^{-5}	9.6573×10^{-11}
10^7	8.7959×10^{-4}	1.1169×10^{-9}
10^8	0.1215	7.5690×10^{-9}
10^9	1.1494	3.7816×10^{-8}

consistently produces vectors which are more orthogonal than those generated by CGS. When $\kappa_2(\boldsymbol{A})=10^9$, for CGS, significant error has accumulated and it dominates the calculation. On the other hand, the orthogonality among the basis produced by OCA is weakened but not completely destroyed. The experimental results demonstrate that OCA can extract numerical orthonormal components. To ensure the numerical stability, as mentioned in Section 3.3, OCA employs a simple but effective threshold policy.

4.1.2. The issue of threshold function setting

In this section, we take experiments on synthetic dataset to evaluate the performance of OCA with different threshold function settings. The dataset is generated as follows. Firstly, we randomly generate $D\times d_0$ basis matrix $\boldsymbol{W}=[\boldsymbol{w_1},\boldsymbol{w_2},\ldots,\boldsymbol{w_{d_0}}]$, the d_0 columns of $\boldsymbol{W}$ are mutually orthonormal. Then, we compute $\boldsymbol{M_1}=\boldsymbol{WH}$ where each entry of coefficient matrix $\boldsymbol{H}\in\mathbb{R}^{D\times N}$ is generated with the normal distribution (mean 0, variance 1), that is, $\boldsymbol{M_1}=\boldsymbol{W}\times randn(d_0,N)$. After that, we compute the average value of the entries of $\boldsymbol{M_1}$: $x_m=(1/DN)\sum_{i,j}|M_1(i,j)|$, add Gaussian noise proportional to x_m: we generate $\boldsymbol{M_2}=\delta x_m\times randn(D,N)$ where $\delta=0.02$ is the noise level, and finally obtain data matrix $\boldsymbol{X}=\boldsymbol{M_1}+\boldsymbol{M_2}$ that contains N D-dimensional data vectors.

As described above, the first component learned by OCA is the longest data vector in the dataset. As a forward selection algorithm, the previous learning steps influence the following steps. As a result of that, we also conduct a series of experiments in the condition that the firstly selected vector is strongly independent with the other vectors in the dataset. In these experiments, we add an extra vector $\boldsymbol{x_0}$ to the previously generated dataset. $\boldsymbol{x_0}$ is λ times as long as the longest data vector among the N vectors $\{\boldsymbol{x_i}\}_{i=1}^{N}$ and it is orthogonal to $S_0=span\{\boldsymbol{w_1},\boldsymbol{w_2},\ldots,\boldsymbol{w_{d_0}}\}$.

Suppose OCA is employed on the synthetic dataset ($f(\omega)$ is the threshold function) and obtains feature subspace $S=span\{\boldsymbol{b_1},\boldsymbol{b_2},\ldots,\boldsymbol{b_d}\}$ where d is the estimated intrinsic dimension of this dataset. To evaluate the quality of the obtained S, we calculate the square of the distance from d_0-dimensional subspace S_0 to d-dimensional subspace S by:

$$Dist^2=d_0-\sum_{i=1}^{\min(d_0,d)}\cos^2\alpha_i=d_0-\sum_{i=1}^{d_0}\sum_{j=1}^{d}(\boldsymbol{w_i}^{\top}\boldsymbol{b_j})^2 \tag{21}$$

where α_i is the i^{th} principal angle between subspaces S_0 and S. Obviously, when d is fixed, the smaller $Dist^2$ is, the better the quality of the learned S is. When $Dist=0$, we have $S_0\subseteq S$.

The experimental results are reported in Table 3. They are the average of 10 replicates. We set $N=200$ in all these experiments.

According to (11), we can obtain the obvious conclusions: the stricter the employed threshold function $f(\omega)$ is, the smaller d the dimension of the subspace obtained by OCA is; the larger $r_{1,1}$ is, the smaller the estimated d is. The results in Table 3 are consistent with the above conclusion.

Table 3
The results of experiments on synthetic data. N is the number of original data that generated by d_0 basis, d_0 is the intrinsic dimension of the dataset $\{\boldsymbol{x_i}\}_{i=1}^{N}$ whose size is N. " – " means that there is no outlier $\boldsymbol{x_0}$ in the dataset, λ means that the $\boldsymbol{x_0}$ outlier is λ times as long as $\max\{\|\boldsymbol{x_1}\|_2,\|\boldsymbol{x_2}\|_2,\ldots,\|\boldsymbol{x_N}\|_2\}$. $f(\omega)$ is the employed threshold function. d is the dimension of feature subspace S learned by OCA. $Dist^2$ is the square of the distance from S_0 to S.

d_0 & D	λ	$f(\omega)=\sqrt{\omega}$		$f(\omega)=\omega$		$f(\omega)=\omega^2$	
		d	$Dist^2$	d	$Dist^2$	d	$Dist^2$
$d_0=10$	–	9.2	0.8012	10	0.0016	10	0.0016
$D=30$	$\lambda=2$	5.2	5.8003	9.4	1.6010	11	0.0015
	$\lambda=5$	2	9.0001	5.1	5.9003	10.1	0.9012
	$\lambda=10$	1	10	3	8.0001	8.2	2.8007
$d_0=20$	–	18.7	1.3058	20	0.0063	20	0.0063
$D=100$	$\lambda=2$	11.8	9.2017	20.4	0.6064	21	0.0072
	$\lambda=5$	3.8	17.2003	13.1	7.9021	21	0.0072
	$\lambda=10$	1	20	8.1	12.9009	20.4	0.6064

In the cases without $\boldsymbol{x_0}$, when $f(\omega)=\omega$ or $f(\omega)=\omega^2$ is employed, OCA is able to extract d_0-dimensional subspace accurately: $d=d_0$ and $Dist^2$ is very small ($Dist^2\neq0$ is due to the additive noise). By comparison, while employing $f(\omega)=\sqrt{\omega}$ as the threshold function, OCA fails to learn all the d_0 components. This means as a threshold function, $f(\omega)=\sqrt{\omega}$ is too strict.

When $\boldsymbol{x_0}$ is added to the dataset, as λ increases, reconstructing S_0 becomes more and more difficult. However, the performance of OCA with $f(\omega)=\omega^2$ is still acceptable: when $\lambda=2$ or $\lambda=5$, OCA can find all orthogonal components. At a rough estimate, when $d_0=20, D=100$ and $\lambda=10$, it extracts more than 20 components and $Dist^2$ is not large. Note that the variance in the direction of $\boldsymbol{x_0}$ is about $\lambda^2\sum_{i=1}^{d_0}z_{0,i}^2$, while the variance in the direction of $\boldsymbol{v_j}(j=1,2,\ldots,d_0)$ is about $\sum_{j=1}^{N}z_{j,i}^2$ ($z_{i,j}$ is an entry with the normal distribution). When $N=200, d_0=10$ and $\lambda=10$, the former is roughly 5 times as large as the latter; when $N=200, d_0=20$ and $\lambda=10$, the former is roughly 10 times as large as the latter. Therefore, at this time, it is easy for DR algorithms to omit the information in the direction of $\boldsymbol{v_j}$. When $r_{1,1}=\|\boldsymbol{x_0}\|_2$ is large, the independence of the other data relatively becomes weaker. Without any prior knowledge, it is natural for OCA to estimate a lower target dimension. Note that as a universal algorithm, OCA is not 0designed for eliminating outliers. In some applications, to eliminate the effects of the too large $r_{1,1}$, user can preprocess the original data by normalization operation.

The learning process of OCA is a trade-off: on one hand, the extracted feature subspace tries to contain all the original data; on the other hand, the dimension of the feature subspace should be the smaller the better. Though we hope the estimated target dimension d is optimal. Suppose D is the dataset's original dimension and d_0 is its intrinsic dimension. When $r_{1,1}$ is abnormally large or d_0 is slightly smaller than D, the estimated d may be smaller than the true intrinsic dimension d_0, as OCA may automatically trend to preventing the expansion of feature subspace. Due to the results listed in Table 3, to extract more information in the above conditions, we believe selecting $f(\omega)=\omega^2$ as the default threshold function is a not bad choice and it is employed in the following experiments. Note that for different applications, users can also set $f(\omega)$ by themselves.

4.2. Experiments on real-world data

To illustrate that OCA can achieve a sound learning performance, we also employ some real-world datasets from [57,58]. These datasets have been widely used to testify the performance

of dimensionality reduction algorithms. The information of the datasets is listed in Table 4. To demonstrate that OCA is a fast component learning algorithm that can deal with large scale data, among them there are three large datasets: Arcene, IJCNN1 and Sensit Vehicle. For database Arcene, the dimension of original data is large; for datasets IJCNN1 and Sensit Vehicle, the size of training data is large.

4.2.1. Performance evaluation on real-word dataset.

In this section, we try to validate the efficiency of the adaptive threshold scheme in the real-world datasets. We compare OCA with several cases that the target dimension is predetermined. For those cases, instead of using the threshold scheme to automatically learn the target dimension, we artificially predetermine the target dimension as 10%, 25%, 50%, 75%, and 100% of the input data dimension D respectively.

The experiment is taken on Hill, Ionosphere, Optical Digits, Sonar and Waveform datasets. We adopt the learned components to transform the original data into low-dimensional data. In the obtained feature space, 1-nearest neighbor (1-NN) rule is employed to classify the test data. Fig. 3 shows the recognition rate with different artificially predetermined components number. The recognition performance under the case of OCA (with threshold scheme) is highlighted by a blue star.

Table 4
Overview of the datasets used in experiments.

Dataset	Training samples	Test samples	Original dimension	Classes
Arcene	100	100	10,000	2
Hill	606	606	100	2
IJCNN1	49,990	91,701	22	2
Ionosphere	316	35	34	2
Optical Digits	3823	1797	64	10
Sensit Vehicle	78,823	19,705	50	3
Sonar	100	108	60	2
Waveform	300	5000	21	3

Fig. 3 shows that the target dimension obtained by OCA in different datasets is different. Therefore, OCA is data-dependent. The optimal compression ratio (CR) varies drastically under different datasets. Though OCA cannot always find the optimal number of components, the adaptive threshold scheme ensures OCA never to achieve a too small target dimension. Note that the adaptive threshold scheme employed by OCA is simple and the computational cost of threshold judging is very low, but it helps OCA obtain better recognition performance in Hill and Ionosphere datasets, and in the other three datasets, the performance of OCA is not bad. As a result of that, we can declare that as a linear dimensionality algorithm, OCA can estimate dataset's intrinsic dimension efficiently and effectively.

4.2.2. Comparison with typical algorithms

In this section, PCA, BPCA [36], CCIPCA [33] and FastICA [35] are employed to make a comparison with OCA. PCA and BPCA both have to solve eigenproblem in their dimensionality reduction process. Although FastICA is a fast version of ICA, usually it employs eigenvalue decomposition for whitening. BPCA is able to estimate the target dimension, but its computational cost is great. CCIPCA applies iterative learning scheme to avoid solving eigenproblem. The characteristics of these algorithms are summarized in Table 5.

Firstly, we take experiments in all datasets listed in Table 4 and report the recognition performance of OCA and the typical algorithms.

It must be emphasized that the target dimension of OCA and BPCA is automatically estimated during the basis learning. But for

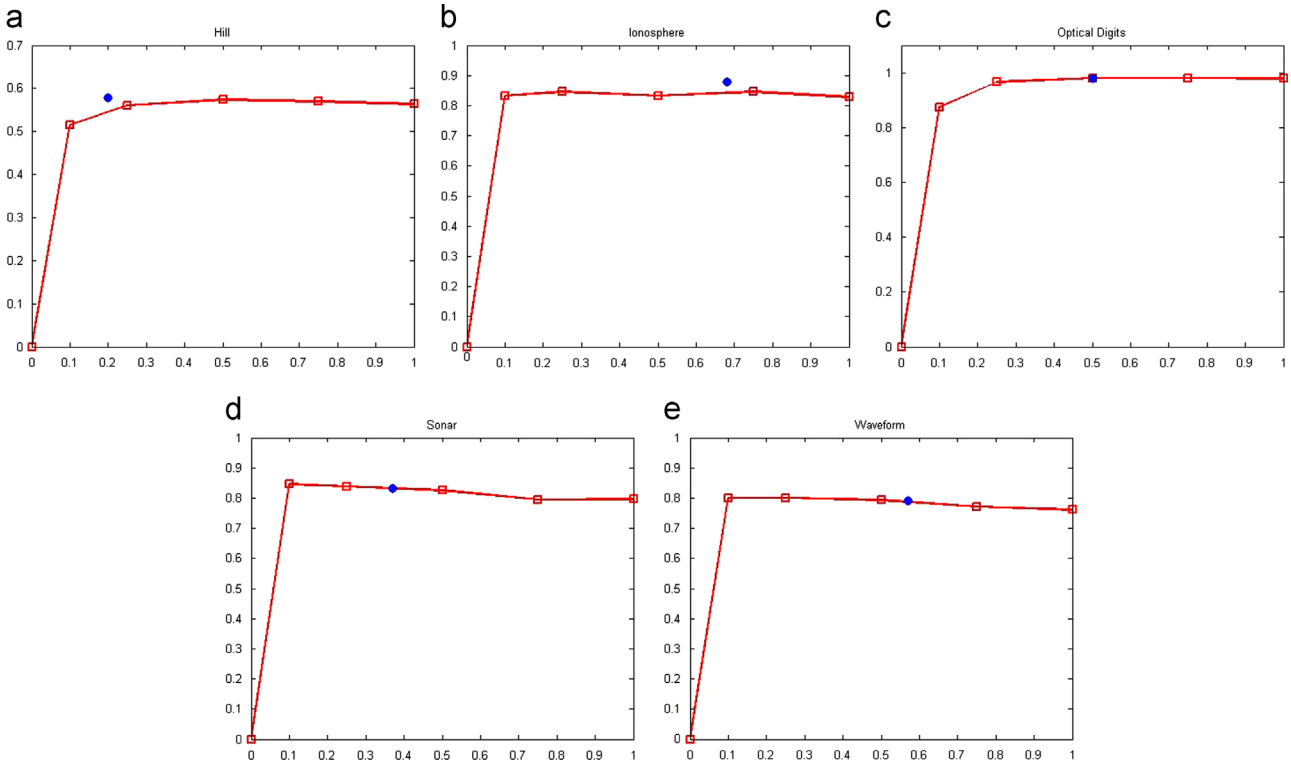

Fig. 3. The recognition rate under different artificially determined components number: the abscissa stands for the compression ratio and the vertical ordinate stands for the corresponding recognition rate. The condition of OCA with artificially predetermined target dimension is highlighted with a red solid line. For OCA with automatically learned target dimension, the recognition rate and the compression ratio is marked with one blue point to make a comparison. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

PCA, CCIPCA, and FastICA, we have to predetermine the target dimension before executing the algorithms. For these algorithms, we tune the target dimension by ten times 10-fold cross validation of training set to achieve an optimal recognition performance. Therefore, we are able to give the recognition result directly. The automatically generated dimension of OCA and BPCA and the tuned basis dimension for PCA, CCIPCA, and FastICA are listed in Table 6. *CR* of such algorithms are also reported.

For each of the eight datasets, after all the original data are transformed into low-dimensional feature space, 1-nearest neighbor (1-NN) rule is used to classify the test data. The recognition results are shown in Table 7. Because the basis vectors obtained by OCA are normalized, we also normalize the components obtained by other algorithms for the purpose of fair comparison. We also list the recognition rate that is obtained merely by NNC without dimensionality reduction as a benchmark.

From Table 7 it is apparent that in almost all the datasets, OCA achieves the best recognition performance. In datasets Optical Digits and Sensit Vehicle, PCA works a little better than OCA. However, from Table 6 we know that, in these two cases, PCA needs more components than OCA. And for all these eight datasets, OCA obtains the best average recognition performance. Therefore, OCA is able to obtain higher recognition rate than PCA, BPCA, CCIPCA, and FastICA. Moreover, Table 6 indicates that sometimes OCA needs fewer components than other methods while its recognition performance is better. For IJCNN1, Ionosphere, Optical Digits, Sensit Vehicle, and Waveform, OCA achieves the best compression ratio.

In Arcene, the dimension of original data D is greatly larger than the size of training set N. In this case, according to the adaptive threshold scheme, the target dimension automatically obtained by OCA is equal to the training set size 100. Meanwhile, due to the great D, BPCA fails to work. For PCA and FastICA, they achieve the optimal result when the target dimension is set to be 100. Although the optimal target dimension for CCIPCA is 89, that is only a little smaller than the training set size. Furthermore, for CCIPCA, it is more suitable to set small target dimension. If the target dimension is set to be too large, the orthogonality among the components learned by CCIPCA will be destroyed and the performance of CCIPCA will be influenced. Therefore, we surmise that the intrinsic dimension of dataset Arcene will not be much smaller than 100. The target dimension $d=100$ estimated by OCA is not unsuitable.

Tables 6 and 7 , we may draw the conclusion that the proposed OCA has outstanding performance in the aspects of recognition rate and compression ratio. It also should be mentioned that, in these experiments, OCA has great advantage in execution time. Compared with CCIPCA (OCA and CCIPCA have the same time complexity $O(NDd)$), averagely, OCA is about 4.4 times faster than CCIPCA.

For dimensionality reduction issue, the components quality of the learned components is an important factor to evaluate the learning performance. As discussed in Section 3.1, reconstruction cost can be used to evaluate the components quality. Therefore, we employ the mean relative reconstruction cost $\mathcal{E}$ to evaluate the quality of the learned components:

$$\mathcal{E}=\frac{1}{N}\sum_{j=1}^{N}\frac{\|\boldsymbol{x_j}-span\{\boldsymbol{b_1},\boldsymbol{b_2},\ldots,\boldsymbol{b_d}\}\|_2}{\|\boldsymbol{x_j}\|_2} \tag{22}$$

While the mutual orthogonality of the basis $\{\boldsymbol{b_1},\boldsymbol{b_2},\ldots,\boldsymbol{b_d}\}$ is ensured, $\mathcal{E}$ can be rewritten as:

$$\mathcal{E}=\frac{1}{N}\sum_{j=1}^{N}\frac{\|\boldsymbol{x_j}-\sum_{i=1}^{d}\boldsymbol{b_i}\boldsymbol{b_i}^{\top}\boldsymbol{x_j}\|_2}{\|\boldsymbol{x_j}\|_2} \tag{23}$$

For OCA and PCA, we directly employ (23) in their mean reconstruction cost calculation. Though usually the basis obtained by CCIPCA is not ensured to be mutually orthogonal in practice, the basic assumption taken by CCIPCA is that the basis vectors can converge to approximately mutually orthogonal principal components during learning process. As a result of that, we also employ (23) as CCIPCA's mean relative reconstruction cost function. Note that for PCA-based algorithms, their $\mathcal{E}$ are obtained after the data are made to be zero mean.

The $\mathcal{E}$ of PCA, BPCA, CCIPCA, and FastICA is listed to compare with that of OCA in Table 8. For OCA and BPCA, the target dimension can be automatically learned by the algorithms

Table 5
The characteristics for the algorithms: "○" means "Yes"; "×" means "No"; "⊖" means "Yes under certain condition".

Algorithm	OCA	PCA	BPCA	CCIPCA	FastICA
Learn orthogonal components	○	○	×	⊖	×
Data selection	○	×	×	×	×
Determine *d* automatically	○	×	○	×	×
Obviate solving eigenproblem	○	×	×	○	⊖

Table 6
Feature space dimension (d) and compression ratio (CR) for OCA, PCA, BPCA, CCIPCA, and FastICA in datasets: the best performances are emphasized with figures in bold typeface.

Dataset	OCA		PCA		BPCA		CCIPCA		FastICA	
	Dim	*CR*	*d*	*CR*	*d*	*CR*	*d*	*CR*	*d*	*CR*
Arcene	100	0.01	100	0.01	–	–	81	**0.008**	100	0.01
Hill	20	0.20	24	0.24	19	0.19	15	**0.15**	55	0.55
IJCNN1	16	**0.73**	18	0.82	21	0.95	18	0.82	19	0.86
Ionosphere	23	**0.68**	30	0.88	26	0.76	28	0.82	29	0.85
Optical Digits	32	**0.5**	40	0.63	48	0.75	60	0.94	32	0.5
Sensit Vehicle	27	**0.54**	34	0.68	35	0.70	27	0.54	42	0.84
Sonar	22	0.37	29	0.48	19	**0.32**	40	0.67	22	0.37
Waveform	12	**0.57**	14	0.67	15	0.71	16	0.76	12	**0.57**

Table 7
Recognition rate (RR) of experiments: the best and near best performances are emphasized with figures in bold typeface.

Dataset	OCA (%)	PCA (%)	BPCA (%)	CCIPCA (%)	FastICA (%)	NNC (%)
Arcene	**88.0**	**88.0**	–	85.7	**87.8**	88.0
Hill	**57.9**	**57.9**	57.6	57.6	56.2	56.2
IJCNN1	**96.8**	96.0	**96.8**	96.2	95.4	98.8
Ionosphere	**88.0**	87.1	87.4	86.6	86.1	86.1
Optical Digits	98.0	**98.1**	97.9	97.9	98.1	98.0
Sensit Vehicle	67.4	**67.6**	65.3	**67.4**	59.7	67.6
Sonar	**83.3**	82.1	79.8	76.9	77.8	81.5
Waveform	**79.2**	78.3	78.2	76.3	**79.5**	76.1
Average	**82.3**	81.9	–	80.7	80.1	81.5

Table 8
Mean relative cost function $\mathcal{E}$: the components are learned by OCA, PCA, BPCA, CCIPCA, and FastICA respectively.

Dataset	OCA	PCA	BPCA	CCIPCA	FastICA
Hill	0.05	0.05	0.05	0.05	0.71
IJCNN1	0.41	0.17	0.84	0.21	1.36
Ionosphere	0.34	0.27	0.45	1.28	0.83
Optical Digits	0.17	0.18	0.18	0.28	0.63
Sensit Vehicle	0.09	0.09	1.41	0.20	3.32
Sonar	0.08	0.14	0.12	0.73	0.56
Waveform	0.26	0.38	0.72	0.42	0.76
Average	0.20	0.18	0.54	0.45	1.17

themselves. But for PCA, CCIPCA and FastICA, it necessitates that users have to predetermine the target dimension. Therefore, for PCA, CCIPCA and FastICA, we adopt the same target dimension with OCA to guarantee that the comparison is fair. The automatically learned target dimension of OCA and BPCA is listed in Table 6.

In column 2 of Table 8, $\mathcal{E}$ of OCA is listed. Compared with BPCA, CCIPCA and FastICA, the components' quality of OCA is better in almost all datasets. For Optical Digits, Sonar, and Waveform, the cost function of OCA is less than that of PCA. Although in IJCNN1 and Ionosphere, $\mathcal{E}$ of OCA is larger than that of PCA. For all the seven datasets, the average $\mathcal{E}$ of OCA is only a little larger than that of PCA. The objective function of PCA ensures that given a fixed target dimension, PCA obtains the minimum $\sum_{j=1}^{N} \|\boldsymbol{x_j}-\tilde{\boldsymbol{x}}\|_2^2$. However, some original data may be poorly approximated by PCA.

Theoretically, due to the normalize operation, the calculated $\mathcal{E}$ will never be larger than 1. However, in Sensit Vehicle, $\mathcal{E}$ of BPCA is 1.41; in Ionosphere, $\mathcal{E}$ of CCIPCA is 1.28; in IJCNN1 and Sensit Vehicle, $\mathcal{E}$ of FastICA is 1.36 and 3.32 respectively. Obviously, some problems have occurred in the calculations. These algorithms cannot give a definite upper bound of approximate error for each data vector. In these cases, many original data are poorly approximated by the obtained components.

For FastICA, in the beginning, the columns of the un-mixing matrix are initialized randomly. Then they are updated iteratively. For BPCA, it employs the EM algorithm to estimate the model parameters. Both FastICA and BPCA have a serious drawback: they cannot always converge to an ideal steady state. The results in Table 8 show that sometimes the output of BPCA and FastICA reconstructs the original data vectors poorly, especially when the dataset is large.

For CCIPCA, it adopts the assumption that the mutual orthogonality of the learned components is maintained after they have been adjusted. It only focuses on the orthogonality between the newly input data vector and the k learned components. This strategy helps CCIPCA reduce the number of dot products from $k(k+1)/2$ to k in each principal components' update operation. Though in [59], the researchers have theoretically proved that when $N\to\infty$, the vectors obtained by CCIPCA converge to the principal components obtained by PCA, with probability 1. In practical experiments, CCIPCA usually fails to obtain numerical orthogonal components.

With the help of the pivoting operation and the threshold policy, the components obtained by OCA are guaranteed to be orthogonal to each other, not only theoretically but also practically. It is well-known that with the same number of components, orthogonal components preserve more original information than non-orthogonal components. The results in Table 8 support the conclusion that components obtained by OCA have good quality.

Given a dimensionality reduction algorithm, its performance is due to two aspects: (1) whether estimating the data dimension well and (2) whether learning a good low-dimensional representation of the data when the target dimension has been determined. Most of the existing algorithms only focus on the second aspect. By comparison, OCA considers both aspects at the same time. It is more convenient for users and the experimental results demonstrate the superiority of OCA.

5. Conclusion

In this paper, we propose a fast linear dimensionality reduction algorithm named Orthogonal Component Analysis (OCA). Unlike many other dimensionality reduction methods, OCA obviates solving eigenproblem and matrix inversion problem. Therefore, OCA achieves high-speed orthonormal component extraction, and that is a great advantage for OCA in large scale data processing. On the other hand, OCA is numerical stable. By proposing an adaptive threshold policy, the output of the algorithm is quality-ensured. Moreover, thanks to the pivoting procedure and the stopping criterion, OCA is able to estimate the dimension of feature space with no prior knowledge and few additional calculations. Therefore, OCA enjoys high stability and low time complexity.

In the experiments, OCA is compared with several typical linear dimensionality reduction algorithms. The results dictate that in the aspects of recognition performance, compression efficiency and components quality, the performance of OCA is good.

As a universal algorithm, OCA is concise, but it is efficient and effective. It can be employed in various fields. Although OCA is designed as a linear dimensionality reduction algorithm, by combining the power of kernel technique, we believe that it can be employed in non-linear applications.

Acknowledgments

This work partially was supported by the National Natural Science Foundation of China (Grant nos. 61373001, 61375064 and 61373130), Foundation of Jiangsu NSF (Grant no. BK20131279).

References

[1] S. Samarasinghe, Neural Networks for Applied Sciences and Engineering, Auerbach Publications, CRC Press, Boca Raton, Florida, 2007.
[2] H. Cai, K. Mikolajczyk, J. Matas, Learning linear discriminant projections for dimensionality reduction of image descriptors, IEEE Trans. Pattern Anal. Mach. Intell. 33 (2) (2011) 338–352.
[3] Y.Y. Lin, T.L. Liu, C.S. Fuh, Multiple kernel learning for dimensionality reduction, IEEE Trans. Pattern Anal. Mach. Intell. 33 (6) (2011) 1147–1160.
[4] S.P. Crain, K. Zhou, S.H. Yang, H.Y. Zha, Dimensionality reduction and topic modeling: from latent semantic indexing to latent Dirichlet allocation and beyond, Mining Text Data (2012) 129–161.
[5] Y. Halpern, S. Horng, L.A. Nathanson, N.I. Shapiro, A comparison of dimensionality reduction techniques for unstructured clinical text, in: ICML 2012 Workshop on Clinical Data Analysis, 2012.
[6] H. Jgou, F. Perronnin, M. Douze, et al., Aggregating local image descriptors into compact codes, IEEE Trans. Pattern Anal. Mach. Intell. 34 (9) (2012) 1704–1716.
[7] L. Zhuo, B. Cheng, J. Zhang, A comparative study of dimensionality reduction methods for large-scale image retrieval, Neurocomputing 141 (2014) 202–210.
[8] A. Phinyomark, H. Hu, P. Phukpattaranont, C. Limsakul, Application of linear discriminant analysis in dimensionality reduction for hand motion classification, Meas. Sci. Rev. 12 (3) (2012) 82–89.
[9] E.N. Sathishkumar, K. Thangavel, T. Chandrasekhar, A novel approach for single gene selection using clustering and dimensionality reduction, Int. J. Sci. Eng. Res. 4 (5) (2013) 1540–1545.
[10] P.A. Devijver, J. Kittler, Pattern Recognition: a Statistical Approach, Prentice-Hall International, HemelHemstead, Hertfordshire, 1982.
[11] P.N. Belhumeur, J.P. Hespanha, D. Kriegman, Eigenfaces vs. fisherfaces: recognition using class specific linear projection, IEEE Trans. Pattern Anal. Mach. Intell. 19 (7) (1997) 711–720.
[12] P. Comon, Independent component analysis, a new concept? Signal Process. 26 (3) (1994) 287–314.
[13] D. Achlioptas, Database-friendly random projections, in: Proceedings of the Twentieth ACM SIGMOD–SIGACT–SIGART Symposium on Principles of Database Systems, 2001, pp. 274–284.

[14] X.F. He, P. Niyogi, Locality preserving projections, in: Advances in Neural Information Processing Systems, vol. 16, MIT Press, Cambridge, Massachusetts, 2004.
[15] J.P. Ye, Characterization of a family of algorithms for generalized discriminant analysis on undersampled problems, J. Mach. Learn. Res. 6 (2005) 483–502.
[16] D. Cai, X. He, J. Han, H.J. Zhang, Orthogonal Laplacian faces for face recognition, IEEE Trans. Image Process. 15 (2006) 3608–3614.
[17] Z. Yang, E. Oja, Linear and nonlinear projective nonnegative matrix factorization, IEEE Trans. Neural Netw. 21 (5) (2010) 734–749.
[18] H. Yin, Nonlinear dimensionality reduction and data visualization: a review, Int. J. Autom. Comput. 4 (3) (2007) 294–303.
[19] S. Yan, D. Xu, B. Zhang, H.J. Zhang, Q. Yang, S. Lin, Graph embedding and extensions: a general framework for dimensionality reduction, IEEE Trans. Pattern Anal. Mach. Intell. 29 (1) (2007) 40–51.
[20] T.F. Cox, M.A. Cox, Multidimensional Scaling, Chapman and Hall, London, 2001.
[21] J.B. Tenenbaum, V. de Silva, J.C. Langford, A global geometric framework for nonlinear dimensionality reduction, Science 290 (2000) 2319–2323.
[22] T. Kohonen (Ed.), Self-Organizing Maps, 2nd Edition, Springer, Berlin, Germany, 1997.
[23] S.T. Roweis, L.K. Saul, Nonlinear dimensionality reduction by locally linear embedding, Science 290 (2000) 2323–2326.
[24] M. Belkin, P. Niyogi, Laplacian eigenmaps for dimensionality reduction and data representation, Neural Comput. 15 (2002) 1373–1396.
[25] Z. Zhang, H. Zha, Principal manifolds and nonlinear dimensionality reduction via tangent space alignment, J. Shanghai Univ. (English Edition) 8 (4) (2004) 406–424.
[26] W.L. Huang, H.J. Yin, On nonlinear dimensionality reduction for face recognition, Image Vis. Comput. 30 (2012) 355–366.
[27] C. Chen, L.J. Zhang, J.J. Bu, C. Wang, W. Chen, Constrained Laplacian eigenmap for dimensionality reduction, Neurocomputing 73 (2010) 951–958.
[28] Y. Goldberg, A. Zakai, D. Kushnir, Y. Ritov, Manifold learning: the price of normalization, J. Mach. Learn. Res. 9 (2008) 1909–1939.
[29] B. Scholköpf, A. Smola, Müller, Nonlinear component analysis as a kernel eigenvalue problem, Neural Comput. 10 (1998) 1299–1319.
[30] F. Tang, R. Crabb, H. Tao, Representing images using nonorthogonal haar-like bases, IEEE Trans. Pattern Anal. Mach. Intell. 29 (12) (2007) 2120–2134.
[31] A. Hyvarinen, E. Oja, Independent component analysis: algorithms and applications, Neural Netw. 13 (4) (2000) 411–430.
[32] J. Chien, B.C. Chen, A new independent component analysis for speech recognition and separation, IEEE Trans. Audio Speech Lang. Process. 14 (4) (2006) 1245–1254.
[33] J. Weng, Y. Zhang, W.S. Hwang, Candid covariance-free incremental principal component analysis, IEEE Trans. Pattern Anal. Mach. Intell. 25 (8) (2003) 1034–1040.
[34] S. Bartelmaos, K. Abed-Meraim, Fast principal component extraction using givens rotations, IEEE Signal Process. Lett. 15 (2008) 369–372.
[35] I. Dagher, R. Nachar, Face recognition using ipca–ica algorithm, IEEE Trans. Pattern Anal. Mach. Intell. 28 (6) (2006) 996–1000.
[36] C. Bishop, Bayesian pca, In: Proceedings of Advances Neural Information Processing Systems, 1999, pp. 382–388.
[37] C.M. Bishop, Variational principal components, In: 9th International Conference on Artificial Neural Networks, 1999, pp. 509–514.
[38] D. Cai, X. He, Orthogonal locality preserving indexing, In: Proceedings of ACM SIGIR Conference on Research and Development in Information Retrieval, 2005, pp. 3–10.
[39] H. Liu, L. Yu, Toward integrating feature selection algorithms for classification and clustering, IEEE Trans. Knowl. Data Eng. 17 (4) (2005) 491–502.
[40] Y. Xu, F. Shen, J. Zhao, O. Hasegawa, To obtain orthogonal feature extraction using training data selection, International Conference on Information & Knowledge Management (CIKM) (2009) 1819–1822.
[41] G.H. Golub, C.F.V. Loan, Matrix Computations, 3rd ed., Johns Hopkins University Press, Baltimore and London, Baltimore, Maryland, 1996.
[42] S.J. Leon, Å Björck, W. Gander, Gram–Schmidt orthogonalization: 100 years and more, Numer. Linear Algebra Appl. 20 (3) (2013) 492–532.
[43] J. Hua, W. Tembe, E.R. Dougherty, Feature selection in the classification of high-dimension data, in: IEEE International Workshop on Genomic Signal Processing and Statistics, 2008, pp. 1–2.
[44] X. Jin, A. Xu, R. Bie, P. Guo, Machine learning techniques and chi-square feature selection for cancer classification using sage gene expression profiles, Lect. Notes Comput. Sci. 3916 (2006) 106–115.
[45] H. Peng, F. Long, C. Ding, Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy, IEEE Trans. Pattern Anal. Mach. Intell. 27 (2005) 1226–1238.
[46] I.A. Gheyas, L.S. Smith, Feature subset selection in large dimensionality domains, Pattern Recognit. 43 (2010) 5–13.
[47] U. Araújo, B. Saldanha, R. Galvao, T. Yoneyama, H. Chame, H. Visani, The successive projections algorithm for variable selection in spectroscopic multicomponent analysis, Chemom. Intell. Lab. Syst. 57 (2) (2001) 65–73.
[48] H. Ren, C.I. Chang, Automatic spectral target recognition in hyperspectral imagery, IEEE Trans. Aerosp. Electron. Syst. 39 (4) (2003) 1232–1249.
[49] N. Gillis, S.A. Vavasis, Fast and robust recursive algorithms for separable nonnegative matrix factorization, IEEE Trans. Pattern Anal. Mach. Intell. 36 (4) (2014) 698–714.
[50] M.D. Choi, Tricks or treats with the Hilbert matrix, Am. Math. Mon. 90 (1983) 301–312.
[51] X. He, Numerical dependence theorem and its application (in chinese),, Numer. Math. A J. Chin. Univ. 1 (1) (1979) 11–19.
[52] P.A. Businger, G.H. Golub, Linear least squares solutions by householder transformations, Numer. Math. 7 (1965) 269–276.
[53] F. Camastra, Data dimensionality estimation methods: a survey, Pattern Recognit. 36 (2003) 2945–2954.
[54] K. Fukunaga, Intrinsic dimensionality extraction, classification, pattern recognition and reduction of dimensionality, in: Handbook of Statistics, vol. 2, 1982, pp. 347–362.
[55] A. Bjorck, Solving linear least squares problems by Gram–Schmidt orthogonalization, BIT Numer. Math. 7 (1) (1967) 1–21.
[56] Å Björck, Gramcschmidt Orthogonalization: 100 Years and More, IWMS, Shanghai, 2010.
[57] C.L. Blake, C.J. Merz, UCI repository of machine learning databases, University of California Department of Information, Irvine, CA, 1996.
[58] M. Duarte, Y.H. Hu, Vehicle classification in distributed sensor networks, J. Parallel Distrib. Comput. 64 (7) (2004) 826–838.
[59] Y. Zhang, J. Weng, Convergence analysis of complementary candid incremental principal component analysis, in: Technical Report MSU-CSE-01-23, Department of Computer Science and Engineering, Michigan State University, East Lansing, 2011.

Tao Zhu received the master's degree from Department of Computer Science and Technology, Nanjing University, China, in 2012. Currently he is a Ph.D. student at Nanjing University. His research interests include dimensionality reduction, face recognition and neural computing.

Ye Xu received the master's degree from Department of Computer Science and Technology, Nanjing University, China, in 2009. Currently he is a Ph.D. student in Computer Science Department at Dartmouth College. His research interests include social network analysis, dimensionality reduction and feature selection, Bayesian learning, multi-instance learning and multi-label learning, neural networks, human activity modeling and so on.

Furao Shen received the Engineering degree from Tokyo Institute of Technology, Tokyo, Japan, in 2006. Currently he is a professor at Nanjing University, China. His research interests include neural computing and robotic intelligence.

Jinxi Zhao received the Ph.d from Nanjing University, China. Currently he is a professor at Nanjing University, China. His research interests include numerical analysis and neural computing.